A COMPUTATIONAL FRAMEWORK FOR GENERATING SIZING FUNCTION IN ASSEMBLY MESHING

William Roshan Quadros[*], Ved Vyas[*], Mike Brewer[+], Steven James Owen[+], Kenji Shimada[*]

[*]Dept. of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, 15213, USA

[+]Sandia National Laboratories[1], Albuquerque, NM, 871850, USA

ABSTRACT

This paper proposes a framework for generating sizing function in meshing assemblies. Size control is crucial in obtaining a high-quality mesh with a reduced number of elements, which decreases computational time and memory use during mesh generation and analysis. This proposed framework is capable of generating a sizing function based on geometric and non-geometric factors that influence mesh size. The framework consists of a background octree grid for storing the sizing function, a set of source entities for providing sizing information based on geometric and non-geometric factors, and an interpolation module for calculating the sizing on the background octree grid using the source entities. Source entities are generated by performing a detailed systematic study to identify all the geometric factors of an assembly. Disconnected skeletons are extracted and used as tools to measure 3D-proximity and 2D-proximity, which are two of the geometric factors. Non-geometric factors such as user-defined size and pre-meshed entities that influence size are also addressed. The framework is effective in generating a variety of meshes of industry models with less computational cost.

Keywords: Assembly meshing, finite element mesh sizing function, skeleton, and pre-mesh

---

[1] Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000

1. INTRODUCTION

Most automatic, unstructured finite element (FE) meshing algorithms do not recognize the geometric complexity and other non-geometric factors in assembly meshing. Hence, it is difficult to generate a desired 3D mesh of assemblies in one step, and it is worthwhile to split the meshing process into two steps: (1) The analysis of the geometric complexity of the input assembly and other non-geometric factors, and the generation of functions that provide element size, anisotropy, and orientation information; (2) Generation of a FE mesh using the element size, shape and orientation information available from the first step. In this paper, the objective is to analyze the geometric complexity and other non-geometric factors for generating element sizing function for assemblies. This paper does not address element anisotropy and orientation functions.

Sizing function plays a crucial role in mesh generation and in finite element simulation. Meshing assembly models is of increasing importance as simulations are more routinely performed at the system level rather than the part level. Fig. 1 shows a uniform mesh and a graded mesh of a system or assembly. The uniform mesh consists of 39,165 elements and the graded mesh consists of only 17,210 elements, with fine mesh at the holes (see Fig. 1(b)). This shows that with a proper sizing function, a high quality FE mesh with a fewer number of elements can be obtained. With a high-quality mesh, more accurate FE analysis results can be obtained. With a proper sizing function, the number of elements can be greatly reduced; therefore, memory usage and computation time can be greatly reduced during analysis. Consequently, there is great demand for the automatic generation of proper mesh sizing functions for assemblies.

A computational framework must automate the process of generating the mesh sizing function while meeting industry requirements. Automating the sizing function generation is important because manual specification of the sizing information is tedious and time-consuming, and for complex CAD models it may not be practical to specify the size manually. At the same time, the framework should provide the user with options/controls in generating a variety of meshes to satisfy the industry needs. The framework should consider both geometric and non-geometric factors that influence mesh sizing. The framework should generate one common sizing function for all the geometric entities of the assembly. This common sizing function must provide consistent element sizing across 1D, 2D, and 3D entities of the assembly. This framework is independent of the meshing algorithms that it is used in conjunction with. The

next section gives a brief review of the literature on previous work in mesh sizing.

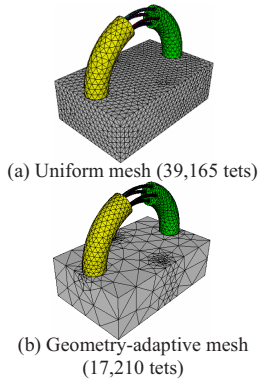


Fig. 1. Uniform and Graded Assembly Meshes

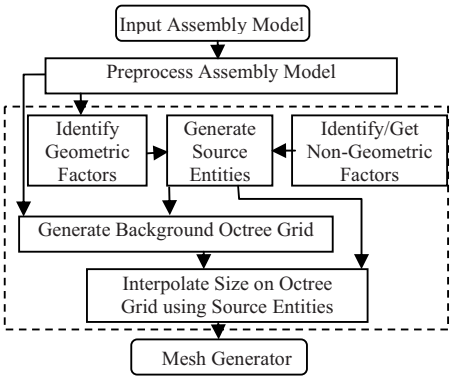


Fig. 2. Schematic Diagram of the Framework

2. LITERATURE REVIEW

This section discusses previous meshing approaches with more emphasis on sizing. Many of the previous meshing algorithms incorporate some sort of element/nodal spacing control; however, sizing is integrated with the meshing process in many cases. Also, consideration has been given to mesh sizing for a single part, not an assembly.

In the past, background meshes have been used as a mechanism for storing sizing function. In early advancing front methods [1], a background mesh consisting of simplicial elements (triangles) was manually constructed using sample points. The generation of a background mesh was later automated by generating Constrained Delaunay Triangulation (CDT) of a set of vertices. Cunha et al. [2] automate the placement of the background mesh nodes on the curves, followed by placement on surfaces, using curvature and proximity. Proximity is determined based on the distance between the facets and the nodes. Measuring proximity in this manner is a combinatorial problem and is generally time-consuming and less accurate. Owen and Saigal [3] use a natural-neighbor interpolation method on a background mesh to alleviate the abrupt variations in target mesh size. In their approach, the sizing function is critically dependent upon node placement in an initial background mesh.

One disadvantage of the background tri/tet mesh is that, while calculating the mesh size at a point, finding the tri/tet that contains the point is expensive. An alternative for storing mesh sizing function is the background grid. Pirzadeh [4] uses a uniform Cartesian grid to store the mesh sizing function. However, a uniform grid is not suitable for capturing large gradients of sizing function, and it consumes a large amount of memory. Another class of background grids that have overcome the uniformed grid limitations are non-uniform hierarchical grids called the "Quadtree" and "Octree" [5, 6]. The size of quadtree/octree cells (squares/cubes) depends on the subdivision of the bounding box, which is governed by the user-specified spacing function or a balance condition for the tree. The drawback of this approach is that quadtree and octree are orientation sensitive, and it is difficult to control the sizing gradient. Zhu et al. [7] use octree as a background overlay grid to store mesh sizing, rather than for the primary purpose of meshing. Their refinement is not directly based on the geometry of the domain, but rather on the sizing function gradient. Zhu [8] has extended the background overlay grid approach to consider pre-meshed geometric entities.

Researchers have also looked at limiting the gradients of sizing functions. Borouchaki et al. [9] present a corrective procedure to control the size gradation. In their method, gradients of a discretized size function are limited by iterating over the edges of a background mesh and updating the size function locally for neighboring nodes. Persson [10] proposes a method for limiting the gradients in a mesh size function by solving a non-linear partial differential equation on the background mesh.

Another class of meshing approach relevant to this paper uses geometric skeletons, in particular medial axis transform (MAT) [11] for geometry-adaptive meshing. Researchers [12] use the radius function of MAT to control nodal spacing on the boundary and interior of a 2D domain while generating a triangular mesh adaptively. Quadros et. al. use the medial axis to generate adaptive quadrilateral meshes on surfaces by varying the width of the tracks using the radius function [13]. They also use skeletons in generating the mesh sizing function for surfaces [14] and single solids [15], by considering only geometric factors.

3. PROBLEM STATEMENT

Given an assembly $\boldsymbol{A}$ *in* R^3 *and the bounds on mesh size,* d_{min}, *and* d_{max}, *and the upper bound on the discrete gradient* α. *Develop a framework for*

automatically generating the mesh sizing function s, based on both geometric and non-geometric factors, such that,

1. *Mesh size* $d = s(\boldsymbol{p})$ *where point* $\boldsymbol{p}(x,y,z) \in \boldsymbol{A}$ *and* $d_{min} \leq d \leq d_{max}$
2. *For any two grid nodes* $\boldsymbol{n_1}$, $\boldsymbol{n_2} \in \boldsymbol{O}$*, where* $\boldsymbol{O}$ *is the background octree grid of* $\boldsymbol{A}$

$$| s(\boldsymbol{n_1}.\text{coord}()) - s(\boldsymbol{n_2}.\text{coord}()) | \leq \alpha \, || \boldsymbol{n_1}.\text{coord}() - \boldsymbol{n_2}.\text{coord}() \, ||$$

4. OVERVIEW OF THE FRAMEWORK

The schematic diagram of the computational framework for generating the mesh sizing function of assemblies is shown inside the dotted lines in Fig. 2. The input assembly consists of many single parts, which are represented in B-rep format (e.g. ACIS sat file, supported by many commercial geometric modelers). The input assembly is preprocessed before passing it to the mesh sizing framework. Then source entities are generated based on the geometric and non-geometric factors to control the size. The source entities provide sizing information, and they are associated with a firmness level to facilitate the overriding of sizes among the different factors. In Section 6.1, a systematic study is performed to identify the geometric factors that must be considered to completely measure the geometric complexity of an assembly. Section 6.2 discusses the non-geometric factors that influence the mesh size. Next, to store the sizing function, a background octree grid is generated, instead of a background mesh, by using both geometric data and source entities (detailed in Section 7). Because a point containment test is expensive in a background mesh, a background grid is used, with a time complexity of O (log N), where N is the number of leaf cells. The final step is to interpolate the sizing function over the background grid using the source entities (detailed in Section 8). The interpolation scheme blends the size smoothly while respecting the firmness levels of geometric and non-geometric factors. During mesh generation, the target mesh size at a point is interpolated using the size stored at the grid-nodes of the octree cell containing that point.

5. PREPROCESS ASSEMBLY MODEL

In the preprocessing stage, an input assembly is made ready for the sizing function module. First, the assembly model is repaired, or healed, if nec-

essary. To obtain a conformal mesh, imprint and merge operations are performed on the assembly model. Note that element sizes are affected by the imprint and merge operations as they change the geometry and topology of a domain at the interfaces. In Fig. 3, at 'A', the circular curve of the hole comes in close proximity with another curve, edge of the cube base. At 'B', the cube shown in green shares a common square interface surface with the circular base, shown in red. As the thickness of the circular base is much less than that of the cube, a fine mesh is required, even at the cube near the interface.

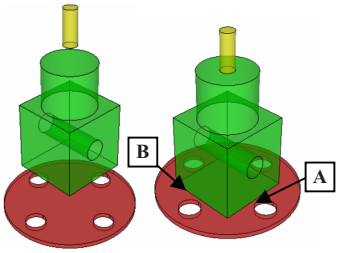


Fig. 3. Before and After Imprint and Merge Operations

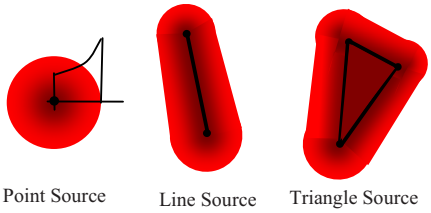


Fig. 4. Point, Line, and Triangle Source Entities

6. GENERATE SOURCE ENTITIES BASED ON GEOMETRIC AND NON-GEOMETRIC FACTORS

A source entity represents the size and gradient at a location due to a geometric or non-geometric factor. Fig. 4 shows source entities that are defined by: coordinate **c**[]; size *s*[]; scope *scp*[]; and local sizing function *f*. A source entity can be a point, a line segment, a triangle, or a tetrahedron, defined by **c**[], *s*[], and *scp*[] (See Fig. 4). The local sizing function *f* can be constant (CONST), linear (LINEAR), geometric progression (GEOM), exponential function (EXP), etc. Source entities are also associated with a firmness level, *l,* and a list of incident geometric entities, *L*. The influence of source entities will be restricted to only the geometric entities present in the list of incident entities, L. The three levels of firmness are LIMP, SOFT, and HARD. The size is imposed only on the geometric entities of *L*. By controlling the size, scope, local sizing function, and firmness level, a variety of meshes can be generated.

6.1. Generate Source Entities Based on Geometric Factors of an Assembly

As it is difficult to analyze the geometric complexity of an assembly, *A*, at once, first the assembly *A,* embedded in $\Re^3$, is decomposed into disjoint subsets; then the geometric complexity of each subset is analyzed in reference to the FE mesh generation. This decomposition of an assembly into disjoint subsets is for the purpose of theoretical analysis only.

6.1.1. Disjoint Subsets of an Assembly

Let an assembly ***A*** contain N solids, $\boldsymbol{S}_i$, where i = 1, 2 … N, which do not intersect. Here it is assumed that the imprinted and merged assembly ***A*** contains curvature-continuous interfaces and bounadries with no degenerate entities.

Fig. 5 shows the anatomy of ***A***, where interior, boundary, and interface are denoted by "*in*", "*bnd*", and "*inf*", respectively. The root of the tree is assembly ***A***, and the leaf nodes are the disjoint subsets, which are shown in rectangular blocks. Fig. 5 shows the disjoint subsets of assembly ***A***, the interior of each solid, $in(\boldsymbol{S}_n)$, the interior of each boundary surface, $in(\boldsymbol{F}^*_m)$, the interior of each interface surface, $in(\boldsymbol{F}^{**}_l)$, the interior of each boundary curve, $in(\boldsymbol{C}^*_p)$, the interior of each interface curve, $in(\boldsymbol{C}^{**}_q)$,and vertices, $\boldsymbol{V}_r$. As the subsets are disjoint, the geometric complexity of each subset is independent of the others.

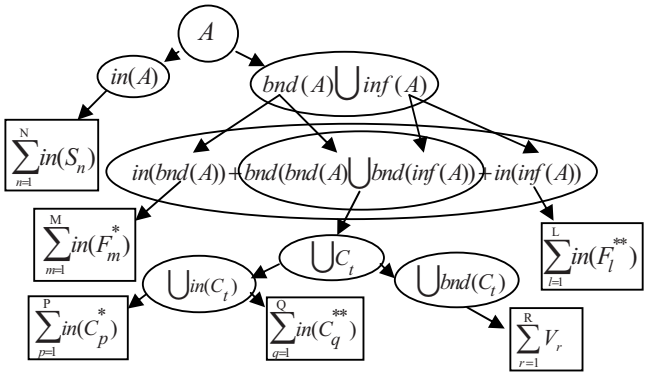


Fig. 5. Disjoint Subsets of an Assembly

6.1.2. Geometric Factors for Each Disjoint Subset

The geometric complexity of each disjoint subset is analyzed in reference to FE meshing to determine the geometric factors [14, 15]. The only geometric factor essential to capture the complexity of the interior of a solid, *in*(***S***), is 3D-proximity. 3D-proximity implies proximity between the vertices, curves, and surfaces that bound *in*(***S***), which is a global measure of the geometric complexity of *in*(***S***). The geometric factors which capture the complexity of the interior of a boundary surface, *in*(***F****), or interface surface, *in*(***F*****), are the 2D-proximity between its boundary curves and vertices, and curviness. 2D-proximity is a global measure, and curviness is a local measure of geometric complexity. The geometric factors of the interior of a boundary curve, *in*(***C****) or interface curve, *in*(***C*****), are the 1D-proximity between end vertices of a curve, curviness and twist [14]. As interior of a curve is a 1D entity embedded in 3D space, these three geometric factors are required in order to completely measure its complexity. No geometric factors are required for a vertex *V*; a FE node at each vertex is sufficient to represent the vertex in a FE model.

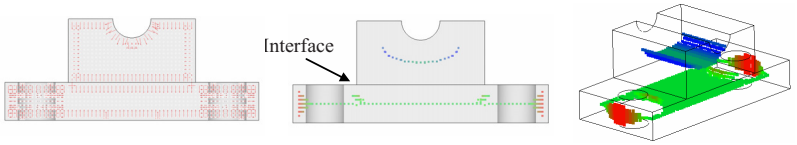
In the beginning of Section 6.1.1 it was assumed that curves and surfaces were curvature continuous, but cusps may exist (such as the tip of cone), and sharp bends in the curves and surfaces, where tangent and curvature vector are not well defined. These points, curves, and regions should be considered as hard points, curves, and regions [14]. These hard entities are not addressed in the remainder of this paper.

6.1.3. Tools to Measure Geometric Factors of Each Subset

The following paragraphs discuss the tools needed to measure the geometric factors of each subset of ***A***. In this paper, disconnected skeletons are proposed as the tools to measure the proximity in *in*(***A***), *in*(***F****), and *in*(***F*****), more accurate tools than previous methods used by researchers.

A 3D-skeleton of an assembly is extracted and used as the tool to measure 3D-proximity in *in*(***A***). On the PR-octree, the 3D-skeleton of an assembly is extracted by propagating a wave front from the boundary surfaces and the interface surfaces towards the interior of an assembly, similar to that of extracting the 3D-skeleton of a single solid [16]. The three important phases of the wave propagation are: initiating the initial wave front (see Fig. 6(a)), propagating the wave front, and terminating the wave. Note that in Fig. 6(a), the wave is propagated in both directions at the interfacial surfaces, whereas at the boundary surfaces the wave is only propagated inward. The skeleton points are generated where the wave

terminates. The distance traveled (also called the radius) by the wave at the skeleton points measures 3D-proximity. In Fig. 6(b), the skeleton points where the wave has traveled the least are shown in red and the furthest in blue.

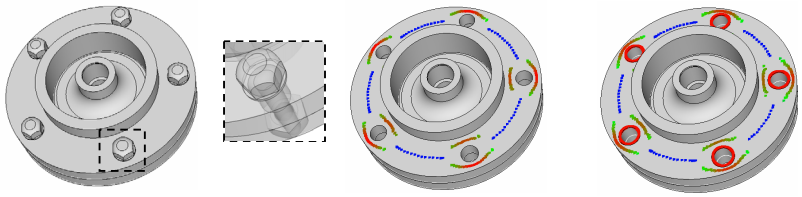


(a) Direction of initial wave front (b) Front view of disconnected 3D-skeleton (c) Isometric view of disconnected 3D-skeleton

Fig. 6. Initial Front and 3D-skeleton

A disconnected skeleton of a surface provides local thickness information, and it is used as a tool to measure the proximity in *in*(***F****) and *in*(***F*****). A disconnected 2D-skeleton of the surfaces is generated by combining the concept of medial axis transform (MAT) and chordal axis transform (CAT) [14]. The skeleton generated using this method is computationally efficient and is sufficiently accurate for the purpose of this work.

Fig. 7 shows how a skeleton effectively measures changes in surface complexity before and after imprint operations. Fig. 7(a) shows the whole assembly model and the magnified view of a bolt and a nut. The 2D-skeleton of the top surface of the flange before the imprinting is shown in Fig. 7(b). Note that after imprinting the circular base of the hexagonal bolt-head on the top surface of the flange, a thin circular ring appears at the holes (Fig. 7(c)). Thus the 2D-skeleton measures the proximity between the outer and inner circles of these thin circular rings.



(a) An assembly model (b) 2D-skeleton before imprinting (c) 2D-skeleton after imprinting

Fig. 7. 2D-skeleton Before and After Imprint

Other tools used to measure other geometric factors mentioned in Section 6.1.2 are briefly discussed here. The curviness in *in*(***F****) and *in*(***F*****) is measured using the minimum principal radius of curvature[2, 3, 14, 17].

The proximity between the end vertices in *in*(***C****) and *in*(***C*****) is measured using the length of a curve. The curviness of *in*(***C****) and *in*(***C*****) is measured using the curvature of a curve [2, 7, 14]. The twist of *in*(***C****) and *in*(***C*****) is measured using torsion [14].

6.1.4. Generate Source Entities using the Tools

Source entities are generated using the tools for controlling the size due to geometric factors. The source entities cover the disjoint subsets in order to reflect the influence of the geometric factors. All the source entities generated based on the geometric factors will be of LIMP firmness level, to give high precedence to non-geometric factors. The following paragraphs explain how to generate the source entities using skeletons and other tools.

A 3D-skeleton is converted into point sources with a size and local sizing function given by Equation (1). The *num_of_layers* controls the number of layers of finite elements across the thickness. A LINEAR local sizing function is used with *end_factor* = 0.1 to have a slightly larger size at the interior of *in*(***S***). The coordinate, **c**, and the scope, *scp*, of a source point is set equal to the skeleton point coordinate and the radius, *R*, respectively. Note that size due to skeleton-based source points are interpolated over the entire grid to cover *in*(***A***).

$$s = \frac{2R}{num\_of\_layers}, f(r) = s - \frac{r}{scp} \times (s \times end\_factor) \tag{1}$$

Triangle sources can be generated using the 2D-skeleton points to control size over *in*(***F****) and *in*(***F*****). The 2D-skeleton radius can be interpolated over the facets (obtained from the geometry engine) to generate triangle sources (Equation 1). A GEOM function is used as local sizing function f, as given in Equation (2) [8]. In Equation (2), r represents the distance between a point **p** (inside the scope) and its projection **p'** on the triangular source. Growth factor g is set to 1.2, and s_0 is the size at **p'**, which is calculated using linear interpolation of sizes on vertices. Thus the size on a source entity is influenced by sizes on the vertices; the sizes at surrounding regions inside the incident entities are controlled by the growth factor. Note that at interfaces *in*(***F*****), the size is radiated inside the incident entities L.

$$f(r) = s_0 \times g^n, n = \frac{\ln\left(\frac{r(g-1)}{s_0} + g\right)}{\ln(g)} \tag{2}$$

The GEOM function is used at both the triangle and the line sources (obtained from facets) on the surfaces and curves, respectively, to radiate

the size into incident entities. In *in*(***F****) and *in*(***F*****), triangular surface curvature-based sources are generated using minimum principal radius of curvature and maximum spanning angles [2, 14, 17] (Fig. 9(c)). In *in*(***C****) and *in*(***C*****) line sources are generated using the tools used for measuring geometric factors of curves [14] (Fig. 9(d)).

6.2. Generate Source Entities Based on Non-Geometric Factors

The following paragraphs discuss non-geometric factors influencing mesh size during simulation of complex assemblies. The firmness level of the source entities can be set depending on the requirement.

User-Defined Size: Source entities are also generated based on the user's input. Experienced users in industry may prefer specific sizes at certain regions based on their knowledge and experience. The framework can handle user-specified sizes on solids, surfaces, curves, and vertices. Users can control the size by specifying size, scope, and local sizing function. Facets of the specified surfaces and curves are extracted, then, based on the user's input, triangle, line, and point sources are generated. The firmness level is set to SOFT to override the sizing functions of geometric factors.

Pre-Meshed Entities: The source entities due to pre-meshed entities are generated to obtain a smooth transition in element size at the pre-meshed entities. Pre-meshed entities often appear while meshing assembly models; they could be surfaces or curves. The 1D/2D elements of the pre-meshed entities are converted into line/triangle sources. Edge lengths of the 1D/2D element are used as size on the source entities, and the growth factor for the GEOM function is set to 1.2. The firmness level is set to SOFT.

Meshing Scheme: Some meshing schemes require a specified number of elements/intervals on the boundary curves. For example, quad meshing algorithms require an even number of 1D elements overall on the boundary curves of a surface. Also, in mapped surface meshing schemes the same number of intervals is required on pairs of opposite curves. These constraints can be imposed by artificially generating the line sources on a curve and by setting the firmness level to HARD.

Previous Analysis Results: Previous analysis results can be incorporated into the sizing function by generating source entities. For example, in a region of stress concentration, the size of source entities can be calculated based on the stress; local sizing function can be controlled based on the

stress gradients, which can then be used in remeshing a model for the next iteration of analysis.

Boundary Conditions: At a boundary condition, the elements' sizes can be controlled before running the analysis by generating source entities using the domain knowledge of the analysis. For example, consider a point load boundary condition, which is frequently encountered in structural/solid mechanics problems. The element size can be controlled using a suitable local sizing function based on radial stress distribution.

7. GENERATE BACKGROUND OCTREE GRID

The background octree grid of an assembly is first generated using geometric information and then refined based on source entities. A PR-Octree is generated using graphics facets because it captures geometric features and provides a suitable lattice for storing a sizing function[15]. The vertices and centroids of facets of an assembly model are given as input point list while generating a PR-Octree starting from the bounding box of the assembly. The graphics facets capture the boundary curvature and small features, and hence small-sized facets exist at high curvature regions and at fine features, which results in finer cells in those regions. The PR-Octree is further subdivided based on the size and sizing gradient of source entities. The cells are further subdivided until only one level of depth difference is maintained between the adjacent cells; this ensures a smooth transition in grid cells for storing the sizing function.

The octree cells intersecting with the boundary facets are efficiently identified using the separating axis theorem [18], and these boundary cells and their nodes are colored gray. Then inner nodes and cells are colored black. Octree cells and nodes lying outside the assembly are removed to reduce memory usage and computational time.

8. INTERPOLATE SIZE ON BACKGROUND GRID USING SOURCE ENTITIES

Interpolating mesh size over the background octree grid using source entities is discussed in two steps. Section 8.1 explains how to interpolate sizing function due to a single factor, geometric or non-geometric. In Section 8.2, the blending of the sizing functions, due to different factors based on firmness level, is explained.

8.1. Interpolate Sizing Function of a Factor

First, every source entity is linked with the grid-nodes that fall inside its scope. The linking process starts from a set of initial nodes of the octree cells that intersect with the source entity. The grid-nodes are visited in breadth-first traversal, starting with the initial nodes, until all the grid-nodes contained inside its scope are linked with the source entity.

The size at a grid node $\boldsymbol{n}$, due to a factor k, is interpolated by taking the weighted sum of the sizes determined by the local sizing function f of m source entities, linked to that grid-node, as given in Equation (3). Here, r_i is the distance between the i^{th} source entity and grid-node $\boldsymbol{n}$, and $f_i(r_i)$ is the size determined by the local sizing function of the i^{th} source entity. k could be any geometric or non-geometric factor. This ensures that a source entity that is closer and has a smaller size has a greater influence on the grid-node.

$$s_k = \sum_{i=1}^{i=m} f_i(r_i) \times \left(\frac{W_{i\_dist} + W_{i\_size}}{2} \right),\ W_{i\_dist} = \frac{\frac{1}{r_i^2}}{\sum_{j=1}^{j=m} \frac{1}{r_j^2}},\ W_{i\_size} = \frac{\frac{1}{f_i(r_i)^2}}{\sum_{j=1}^{j=m} \frac{1}{f_j(r_j)^2}} \tag{3}$$

8.2. Blend Sizing Functions of Geometric and Non-Geometric Factors

This section gives details of blending different sizing functions. The firmness level is used in overriding a sizing function of one factor with that of another. All the sizing functions due to geometric factors have a firmness level of LIMP, and the sizing functions due to non-geometric factors have firmness level of either SOFT or HARD, depending on the firmness level of a source entity (see Section 6.2).

As all the geometric factors have LIMP firmness, the final size s at each grid-node is calculated as given in Equation (4), where k represents different geometric factors. w are the weights, which are initially set to 1.0. These weights can be controlled to achieve variation in meshes. The *scale* is used to control the overall coarseness of the mesh.

$$s = \min\{s_k \cdot w_k\} \cdot scale \tag{4}$$

The sizing function generated by combining all the geometric factors can be overridden by the sizing function of non-geometric factors, as the

non-geometric factors have a higher firmness level. The maximum size, d_{max}, and minimum size, d_{min}, are enforced by trimming the sizing function stored on the background grid.

Smoothing techniques are used to alleviate the abrupt gradients caused by combining the sizing functions of the geometric factors, and by overriding the sizing function due to non-geometric factors. The concepts of digital filters used in smoothing 2D images are used here in smoothing the mesh sizing function stored on the octree. First two iterations of median filter are used to remove the sharp gradients, and then a modified mean filter is used iteratively to smooth the gradients. During mesh generation, trilinear interpolation is used to calculate the target mesh size at **p**, using the size at the grid-nodes of the cell containing **p**. Thus the target mesh size is calculated in $O(max\_depth)$, where *max_depth* is the maximum depth of the octree.

9. RESULTS AND DISCUSSION

The proposed framework has been implemented in C++ within CUBIT, a finite element mesh generation toolkit by Sandia National Laboratories. The proposed framework has been tested on industrial assembly models and results obtained on three such models are shown in Fig. 8 to Fig. 10 (original models courtesy of Ansys, Inc.).

Fig. 9 shows the components of the proposed framework by considering only geometric factors in a twelve-volume assembly. Fig. 9(a) and Fig. 9(b) show 16,037 3D-skeleton points and 16,474 2D-skeleton points, which are used as tools to measure 3D-proximity and 2D-proximity, respectively. Fig. 9(c) and Fig. 9(d) show 7,405 triangle sources and 5,320 line sources on surfaces and curves, respectively. Fig. 9(e) shows the mesh size on the background octree grid of *min_depth* = 5 and *max_depth* = 7 (as in all three models) using a color scale. Fig. 9(f) shows the geometry-adaptive mesh containing 83,584 tet elements.

Fig. 8(b) shows a mesh generated by combining both the geometric factors and sizing based on a pre-meshed surface (see Fig. 8 (a)) in a four-volume assembly. The number of tris in the pre-mesh and average edge length of the tris were 787 and 2.00 respectively. 787 triangle sources were generated using the tris of the pre-meshed surface with sizes calculated based on the edge lengths of tris. A GEOM local sizing function with growth factor = 1.2 was used. The sectional view in Fig. 8(b) shows a smooth transition in the size of the tet elements at the pre-meshed surface. The tet mesh shown in Fig. 8(b) consisted of 44,462 tets.

Fig. 10 shows the meshes generated by combining the geometric factors and a user-defined size in a two-volume assembly. Fig. 10(a) contains 68,325 tets that are generated using geometric factors only. Fig. 10(b) contains 73,289 tets that are generated by incorporating user-defined size specified on the top surface. The user has requested four layers of tet elements of size 2.5. Triangle sources were generated by extracting the ACIS facets with size = 2.5, scope = 10.0, and a CONST local function.

For the assembly in Fig. 9, the maximum and minimum sizes at the grid nodes were 37.20 and 1.29 respectively, which were within the user specified bounds d_{max} = 40, and d_{min} = 1.0. Similarly, maximum and minimum sizes were 15.05 and 1.32, respectively, for Fig. 8. In Fig. 9, maximum and minimum sizes were 40.68 and 1.38, respectively.

Table 1 shows the computational time taken by the components of the framework in generating sizing function for the assembly models shown in Fig. 8 to Fig. 10. The timings were measured in e*machines* M6811 notebook. From Table 1 it is observed that interpolation has taken the most time in all three models. Note that octree generation time in assemblies shown in Fig. 8, Fig. 9, and Fig. 10 is proportional to the number of graphics facets extracted (2,927, 12,161, and 7,346 facets, respectively). There is a strong correlation between number of source entities and computational time for interpolation. The number of source entities in the three models were 11,211, 45,236, and 44,957.

Table 2 shows that the approach used in extracting disconnected 3D and 2D skeletons are computationally efficient. The 2D- and 3D-skeleton times are accompanied by their generation times relative to the total source entity generation times for each assembly. The 2D-skeleton times are much larger than the 3D-skeleton times due to a larger number of surfaces than volumes in the models. For example, the model in Fig. 10 contains only two volumes but has 186 surfaces.

Table 1. Computational Time (sec) in Generating Mesh Sizing Function

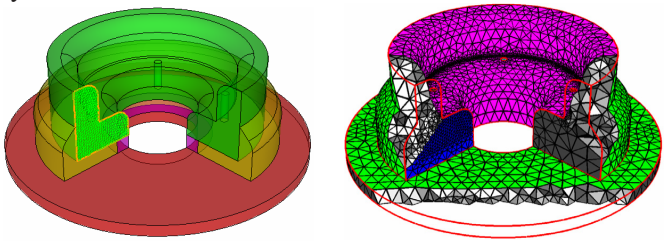
	Figure 8	Figure 9	Figure 10
Octree Generation	1.689	4.438	3.999
Source Entity Generation	0.702	2.547	2.501
Interpolation	2.188	7.453	9.000
Total	4.579	14.438	15.500

Table 2. Computational Time (sec) in Generating Disconnected Skeletons

	Figure 8	Figure 9	Figure 10
3D-Skeleton	0.062 (8.9%)	0.234 (9.2%)	0.282 (11.3%)
2D-Skeleton	0.312 (44.4%)	1.094 (43.0%)	0.782 (31.3%)

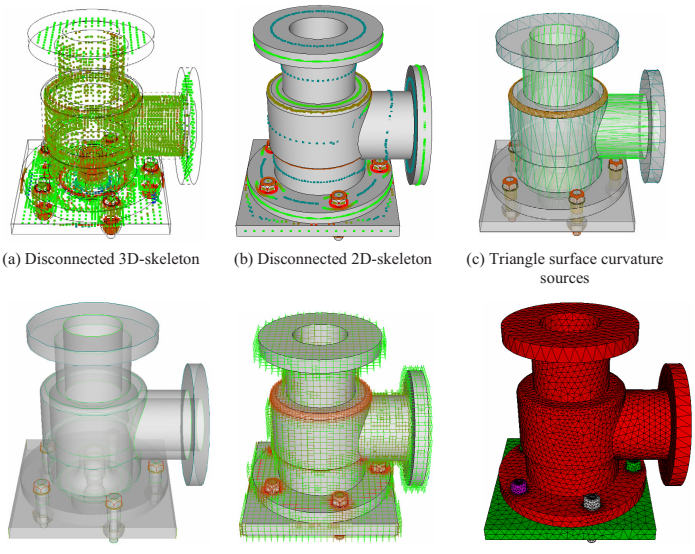
10. CONCLUSION

In this paper a computational framework for generating a mesh sizing function for assembly meshing is proposed. The framework generates a sizing function by considering both geometric and non-geometric factors. A systematic study has been performed to determine the geometric factors that influence the mesh size. Disconnected 3D and 2D skeletons are extracted and used for measuring the proximity. The proposed framework is computationally efficient and it has been tested on many industry models to verify the effectiveness.



(a) Assembly model with a pre-meshed surface (b) Sectional view of a tet mesh generated using geometric factors and a pre-meshed entity

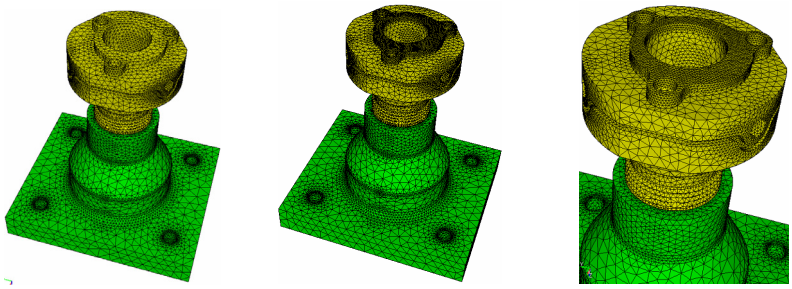
Fig. 8. Mesh Sizing due to Geometric and Non-geometric Factors (pre-meshed entity)



(a) Disconnected 3D-skeleton (b) Disconnected 2D-skeleton (c) Triangle surface curvature sources

(d) Line sources on curves (e) Size on background octree grid (f) Geometry-adaptive mesh

Fig. 9. Components of the Framework



(a) Graded mesh based on only geometric factors

(b) Graded mesh based on geometric factors and a user defined size

(c) Enlarged view of the mesh at a user-defined surface

Fig. 10. Mesh Sizing due to Geometric Factors and a Non-Geometric Factor (user defined size)

REFERENCES

[1] R. Lohner and P. Parikh, "Generation of Three-Dimensional Unstructured Grids by the Advancing Front Method," *AIAA-88-0515*, 1988.

[2] A. Cunha, S. A. Canann, and S. Saigal, "Automatic Boundary Sizing For 2D and 3D Meshes," *AMD Trends in Unstructured Mesh Generation, ASME*, vol. 220, pp. 65-72, 1997.

[3] S. J. Owen and S. Saigal, "Neighborhood Based Element Sizing Control for Finite Element Surface Meshing," *Proceedings, 6th International Meshing Roundtable*, pp. 143-154, 1997.

[4] S. Pirzadeh, "Structured Background Grids for Generation of Unstructured Grids by Advancing-Front Method," *AIAA*, vol. 31, 1993.

[5] W. C. Tracker, "A Brief Review of Techniques for Generating Irregular Computational Grids," *Int. Journal for Numerical Methods in Engineering*, vol. 15, pp. 1335-1341, 1980.

[6] M. S. Shephard, "Approaches to the Automatic Generation and Control of Finite Element Meshes," *Applied Mechanics Review*, vol. 41, pp. 169-185, 1988.

[7] J. Zhu, T. Blacker, and R. Smith, "Background Overlay Grid Size Functions," *Proceedings of 11th International Meshing Roundtable*, pp. 65-74, 2002.

[8] J. Zhu, "A New Type of Size Function Respecting Premeshed Entities," *12th International Meshing Roundtable*, 2003.

[9] H. Borouchaki and F. Hecht, "Mesh Gradation Control," *6th International Meshing Roundtable*, 1997.

[10] P.-O. Persson, "PDE-Based Gradient Limiting for Mesh Size Functions," *Proceedings, 13th International Meshing Roundtable*, pp. 377-388, 2004.

[11] H. Blum, "A Transformation for Extracting New Descriptors of Shape," *Models for the Perception of Speech and Visual Form Cambridge MA The MIT Press*, pp. 326-380, 1967.

[12] V. Srinivasan, L. R. Nackman, J. M. Tang, and S. N. Meshkat, "Automatic Mesh Generation using the Symmetric Axis Transformation of Polygonal Domains," *Proc. IEEE*, vol. 80(9), pp. 1485-1501, 1992.

[13] W. R. Quadros, K. Ramaswami, F. B. Prinz, and B. Gurumoorthy, "Automated Geometry Adaptive Quadrilateral Mesh Generation using MAT," *Proceedings of ASME DETC*, 2001.

[14] W. R. Quadros, S. J. Owen, M. Brewer, and K. Shimada, "Finite Element Mesh Sizing for Surfaces Using Skeleton," *Proceedings, 13th International Meshing Roundtable*, pp. 389-400, 2004.

[15] W. R. Quadros, K. Shimada, and S. J. Owen, "Skeleton-Based Computational Method for Generation of 3D Finite Element Mesh Sizing Function," *Engineering with Computers*, 2004.

[16] W. R. Quadros, K. Shimada, and S. J. Owen, "3D Discrete Skeleton Generation by Wave Propagation on PR-Octree for Finite Element Mesh Sizing," *ACM Symposium on Solid Modeling and Applications*, 2004.

[17] S. J. Owen and S. Saigal, "Surface Mesh Sizing Control," *International Journal for Numerical Methods in Engineering*, vol. 47, pp. 497-511, 2000.

[18] D. Eberly, "Intersection of Convex Objects: The Method of Separating Axes," *Magic Software, Inc.*, 2003.